Machine Learning for NLP
Bonus lecture: The averaged perceptron

UNIVERSITY OF GOTHENBURG

Richard Johansson

September 11, 2015

overview

- a simple modification of the perceptron algorithm
- often gives quite nice improvements in practice

perceptron pseudocode

$\boldsymbol{w} = (0, \ldots, 0)$
repeat N times
  for $(\boldsymbol{x}_i, y_i)$ in training set $\mathcal{T}$
    score $= y \cdot \boldsymbol{w} \cdot \boldsymbol{x}_i$
    if score ≤ 0
      $\boldsymbol{w} = \boldsymbol{w} + y \cdot \boldsymbol{x}$
return $\boldsymbol{w}$

a problem with the perceptron?

- we return the most recent version of the weight vector
- intuitively, this version is over-adapted to the last few instances, and may work less well for other instances

intuition: combining classifiers by voting or averaging

- let's assume we have a lot of classifiers
- each of them has its own strengths and weaknesses
- could they somehow work together?
  - **voting**: return the output favored by most of the classifiers
  - **averaging**: compute the prediction scores for all classifiers; return the output selected by considering the average of all the scores

using averaging to handle the overfitting problem

- in the perceptron, each version of the weight vector can be seen as a separate classifier
  - so we have $N \cdot |\mathcal{T}|$ classifiers
- each of them is over-adapted to the last examples it saw
- but if we compute their average, then maybe we get something that works better overall?
- **averaged perceptron**: return the average of all versions of the weight vector

averaged perceptron pseudocode (naive)

$\mathbf{w}_0 = (0, \ldots, 0)$
$t = 0$
repeat N times
  for $(\mathbf{x}_i, y_i)$ in training set $\mathcal{T}$
    score $= y \cdot \mathbf{w}_t \cdot \mathbf{x}_i$
    if score ≤ 0
      $\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \cdot \mathbf{x}_i$
    else
      $\mathbf{w}_{t+1} = \mathbf{w}_t$
    $t = t + 1$
return $\frac{\mathbf{w}_1 + \ldots + \mathbf{w}_{N \cdot |\mathcal{T}|}}{N \cdot |\mathcal{T}|}$

this is too impractical!

- it's a waste of memory to remember all the versions of $\boldsymbol{w}$ that we have used during training
- can we do something smarter?

an observation

- the weight vector $\mathbf{w}_3$ is the sum of all updates so far:

$$
\begin{aligned}
\mathbf{w}_0 &= (0, \ldots, 0) \\
\mathbf{w}_1 &= \mathbf{w}_0 + \Delta_1 = \Delta_1 \\
\mathbf{w}_2 &= \mathbf{w}_1 + \Delta_2 = \Delta_1 + \Delta_2 \\
\mathbf{w}_3 &= \mathbf{w}_2 + \Delta_3 = \Delta_1 + \Delta_2 + \Delta_3
\end{aligned}
$$

- the average of three vectors can be written:

$$
\begin{aligned}
\frac{\mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3}{3} &= \frac{\Delta_1}{3} + \frac{\Delta_1 + \Delta_2}{3} + \frac{\Delta_1 + \Delta_2 + \Delta_3}{3} \\
&= \frac{3}{3}\Delta_1 + \frac{2}{3}\Delta_2 + \frac{1}{3}\Delta_3
\end{aligned}
$$

better averaged perceptron

$\boldsymbol{w} = (0, \dots, 0)$
$\boldsymbol{a} = (0, \dots, 0)$
$step = N \cdot |\mathcal{T}|$
repeat N times
  for $(\boldsymbol{x}_i, y_i)$ in training set $\mathcal{T}$
    score $= y_i \cdot \boldsymbol{w} \cdot \boldsymbol{x}_i$
    if score ≤ 0
      $\boldsymbol{w} = \boldsymbol{w} + y_i \cdot \boldsymbol{x}_i$
      $\boldsymbol{a} = \boldsymbol{a} + \frac{step}{N \cdot |\mathcal{T}|} y_i \cdot \boldsymbol{x}_i$
    $step = step - 1$
return $\boldsymbol{a}$

in Python

```
class AveragedSparsePerceptron(LinearClassifier):

    # ...

    def fit(self, X, Y):
        # ... initialization ...

        w = numpy.zeros( n_features )
        a = numpy.zeros( n_features )

        NT = self.n_iter * len(Y)
        step = NT

        for i in range(self.n_iter):
            for x, y in zip(X, Yn):
                score = sparse_dense_dot(x, w) * y
                if score <= 0:
                    add_sparse_to_dense(x, w, float(y))
                    add_sparse_to_dense(x, a, step * float(y) / NT)
                step -= 1

        self.w = a
```

experiment (assignment 2 dataset)

- standard: training time 3.4 sec, accuracy 0.809
- averaged: training time 3.6 sec, accuracy 0.829